5.315 stable_compatibility

DESCRIPTION LINKS GRAPH

Origin P. Flener, [39]

Constraint stable_compatibility(NODES)

Argument NODES : collection(index−int, father−dvar, prec−sint, inc−sint)

Restrictions
required(NODES, [index, father, prec, inc])
NODES.index ≥ 1
NODES.index ≤ |NODES|
distinct(NODES, index)
NODES.father ≥ 1
NODES.father ≤ |NODES|
NODES.prec ≥ 1
NODES.prec ≤ |NODES|
NODES.inc ≥ 1
NODES.inc ≤ |NODES|
NODES.inc > NODES.index

Purpose Enforce the construction of a *stably compatible* supertree that is compatible with several given trees. The notion of stable compatibility and its context are detailed in the **Usage** slot.

Example

$$\left\langle \begin{array}{llll} \mathtt{index}-1 & \mathtt{father}-4 & \mathtt{prec}-\{11,12\} & \mathtt{inc}-\emptyset, \\ \mathtt{index}-2 & \mathtt{father}-3 & \mathtt{prec}-\{8,9\} & \mathtt{inc}-\emptyset, \\ \mathtt{index}-3 & \mathtt{father}-4 & \mathtt{prec}-\{2,10\} & \mathtt{inc}-\emptyset, \\ \mathtt{index}-4 & \mathtt{father}-5 & \mathtt{prec}-\{1,3\} & \mathtt{inc}-\emptyset, \\ \mathtt{index}-5 & \mathtt{father}-7 & \mathtt{prec}-\{4,13\} & \mathtt{inc}-\emptyset, \\ \mathtt{index}-6 & \mathtt{father}-2 & \mathtt{prec}-\{8,14\} & \mathtt{inc}-\emptyset, \\ \mathtt{index}-7 & \mathtt{father}-7 & \mathtt{prec}-\{6,13\} & \mathtt{inc}-\emptyset, \\ \mathtt{index}-8 & \mathtt{father}-6 & \mathtt{prec}-\emptyset & \mathtt{inc}-\{9,10,11,12,13,14\}, \\ \mathtt{index}-9 & \mathtt{father}-2 & \mathtt{prec}-\emptyset & \mathtt{inc}-\{10,11,12,13\}, \\ \mathtt{index}-10 & \mathtt{father}-3 & \mathtt{prec}-\emptyset & \mathtt{inc}-\{11,12,13\}, \\ \mathtt{index}-11 & \mathtt{father}-1 & \mathtt{prec}-\emptyset & \mathtt{inc}-\{12,13\}, \\ \mathtt{index}-12 & \mathtt{father}-1 & \mathtt{prec}-\emptyset & \mathtt{inc}-\{13\}, \\ \mathtt{index}-13 & \mathtt{father}-5 & \mathtt{prec}-\emptyset & \mathtt{inc}-\{14\}, \\ \mathtt{index}-14 & \mathtt{father}-6 & \mathtt{prec}-\emptyset & \mathtt{inc}-\emptyset \end{array} \right\rangle$$

Figure 5.561 shows the two trees we want to merge. Note that the leaves `a` and `f` occur in both trees. Figure 5.562 gives one way to merge the two previous trees. This solution corresponds to the ground instance provided by the example. Note that there exist 7 other ways to merge these two trees. They are respectively depicted by Figures 5.562 to 5.569.

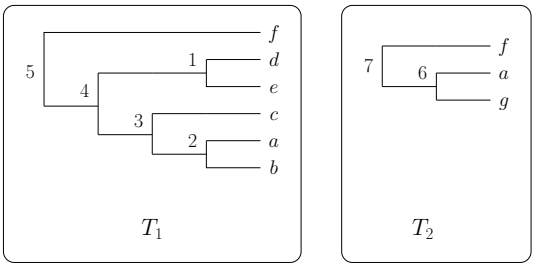

Figure 5.561: The two trees to merge

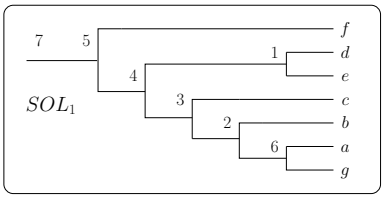

Figure 5.562: First solution corresponding to the ground instance of the example

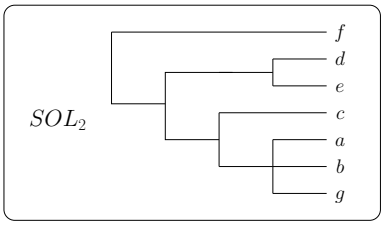

Figure 5.563: Second solution

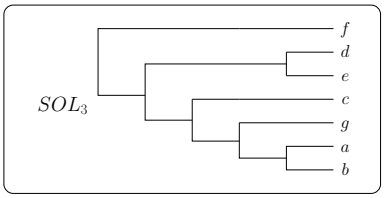

Figure 5.564: Third solution

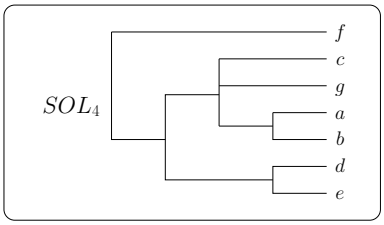

Figure 5.565: Fourth solution

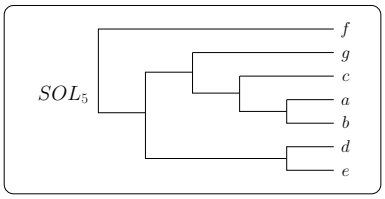

Figure 5.566: Fifth solution

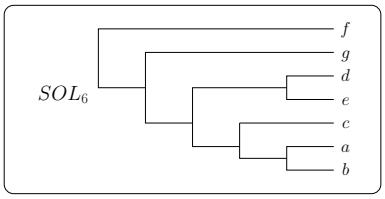

Figure 5.567: Sixth solution

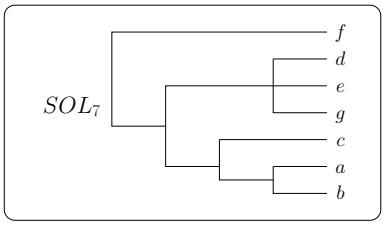

Figure 5.568: Seventh solution

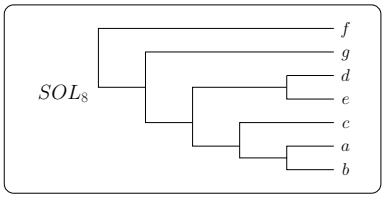

Figure 5.569: Eighth solution

Symmetry Items of NODES are permutable.

Usage One objective of phylogeny is to construct the genealogy of the species, called the *tree of life*, whose leaves represent the contemporary species and whose internal nodes represent extinct species that are not necessarily named. An important problem in phylogeny is the construction of a supertree [69] that is compatible with several given trees. There are several definitions of tree compatibility in the literature:

- A tree $\mathcal{T}$ is *strongly compatible* with a tree $\mathcal{T}'$ if $\mathcal{T}'$ is topologically equivalent to a subtree $\mathcal{T}$ that respects the node labelling. [264]
- A tree $\mathcal{T}$ is *weakly compatible* with a tree $\mathcal{T}'$ if $\mathcal{T}'$ can be obtained from $\mathcal{T}$ by a series of arc contractions. [361]
- A tree $\mathcal{T}$ is *stably compatible* with a set $\mathcal{S}$ of trees if $\mathcal{T}$ is weakly compatible with each tree in $\mathcal{S}$ and each internal node of $\mathcal{T}$ can be labelled by at least one corresponding internal node of some tree in $\mathcal{S}$.

For the supertree problem, strong and weak compatibility coincide if and only if all the given trees are binary [264]. The existence of solutions is not lost when restricting weak compatibility to stable compatibility.

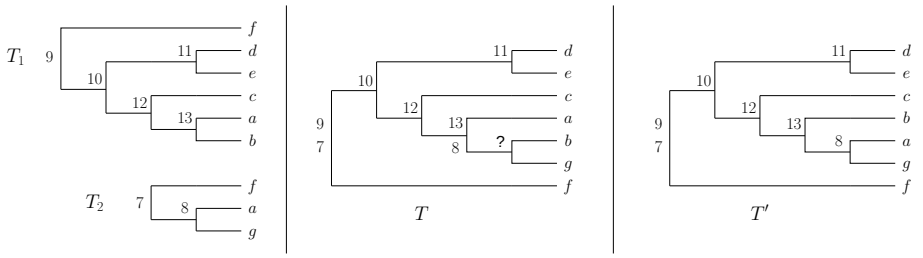

Figure 5.570: Supertree problem instance and two of its solutions

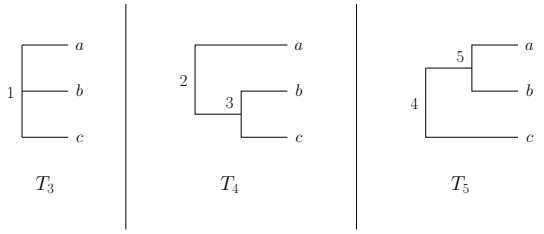

Figure 5.571: Three small phylogenetic trees

For example, the trees $\mathcal{T}_1$ and $\mathcal{T}_2$ of Figure 5.570 have $\mathcal{T}$ and $\mathcal{T}'$ as supertrees under both weak and strong compatibility. As shown, all the internal nodes of $\mathcal{T}'$ can be labelled by corresponding internal nodes of the two given trees, but this is not the case for the father of b and g in $\mathcal{T}$. Hence $\mathcal{T}$ and four other such supertrees are debatable because they speculate about the existence of extinct species that were not in any of the given trees. Consider

also the three small trees in Figure 5.571: $\mathcal{T}_3$ and $\mathcal{T}_4$ have $\mathcal{T}_4$ as a supertree under weak compatibility, as it suffices to contract the arc $(3, 2)$ to get $\mathcal{T}_3$ from $\mathcal{T}_4$. However, $\mathcal{T}_3$ and $\mathcal{T}_4$ have no supertree under strong compatibility, as the most recent common ancestor of b and c, denoted by $mrca(b, c)$, is the same as $mrca(a, b)$ in $\mathcal{T}_3$, namely 1, but not the same in $\mathcal{T}_4$, as $mrca(b, c) = 3$ is an evolutionary descendant of $mrca(a, b) = 2$. Also, $\mathcal{T}_4$ and $\mathcal{T}_5$ have neither weakly nor strongly compatible supertrees.

Under strong compatibility, a first supertree algorithm was given in [3], with an application for database management systems; it takes $O(l^2)$ time, where l is the number of leaves in the given trees. Derived algorithms have emerged from phylogeny, for instance *One-Tree* [264]. The first constraint program was proposed in [169], using standard, non-global constraints. Under weak compatibility, a phylogenetic supertree algorithm can be found in [361] for instance. Under stable compatibility, the algorithm from computational linguistics of [72] has supertree construction as special case.

See also **root concept:** `tree`.

Keywords **application area:** bioinformatics, phylogeny.

constraint type: graph constraint.

final graph structure: tree.

Arc input(s)	NODES
Arc generator	$CLIQUE \mapsto$ collection(nodes1, nodes2)
Arc arity	2
Arc constraint(s)	nodes1.father = nodes2.index
Graph property(ies)	• MAX_NSCC ≤ 1 • NCC $= 1$ • MAX_ID ≤ 2 • PATH_FROM_TO(index, index, prec) $= 1$ • PATH_FROM_TO(index, index, inc) $= 0$ • PATH_FROM_TO(index, inc, index) $= 0$

Graph model

To each distinct leave (i.e., each species) of the trees to merge corresponds a vertex of the initial graph. To each internal vertex of the trees to merge corresponds also a vertex of the initial graph. Each vertex of the initial graph has the following attributes:

- An *index* corresponding to a unique identifier.
- A *father* corresponding to the father of the vertex in the final tree. Since the leaves of the trees to merge must remain leaves we remove the index value of all the leaves from all the father variables.
- A *set of precedence constraints* corresponding to all the arcs of the trees to merge.
- A *set of incomparability constraints* corresponding to the incomparable vertices of each tree to merge.

The arc constraint describes the fact that we link a vertex to its father variable. Finally we use the following six graph properties on our final graph:

- The first graph property $\mathbf{MAX\_NSCC} \leq 1$ enforces the fact that the size of the largest strongly connected component does not exceed one. This avoid having circuits containing more than one vertex. In fact the root of the merged tree is a strongly connected component with one single vertex.
- The second graph property $\mathbf{NCC} = 1$ imposes having only one single tree.
- The third graph property $\mathbf{PATH\_FROM\_TO}(\mathtt{index}, \mathtt{index}, \mathtt{prec}) = 1$ enforces for each vertex i a set of precedence constraints; for each vertex j of the precedence set there is a path from i to j in the final graph.
- The fourth graph property $\mathbf{MAX\_ID} \leq 2$ enforces that the number of predecessors (i.e., arcs from a vertex to itself are not counted) of each vertex does not exceed 2 (i.e., the final graph is a binary tree).
- The fifth and sixth graph properties $\mathbf{PATH\_FROM\_TO}(\mathtt{index}, \mathtt{index}, \mathtt{inc}) = 0$ and $\mathbf{PATH\_FROM\_TO}(\mathtt{index}, \mathtt{inc}, \mathtt{index}) = 0$ enforces for each vertex i a set of incomparability constraints; for each vertex j of the incomparability set there is neither a path from i to j, nor a path from j to i.

Figures 5.572 and 5.573 respectively show the precedence and the incomparability graphs associated with the **Example** slot. As it contains too many arcs the initial graph is not shown. Figures 5.562 shows the first solution satisfying all the precedence and incomparability constraints.

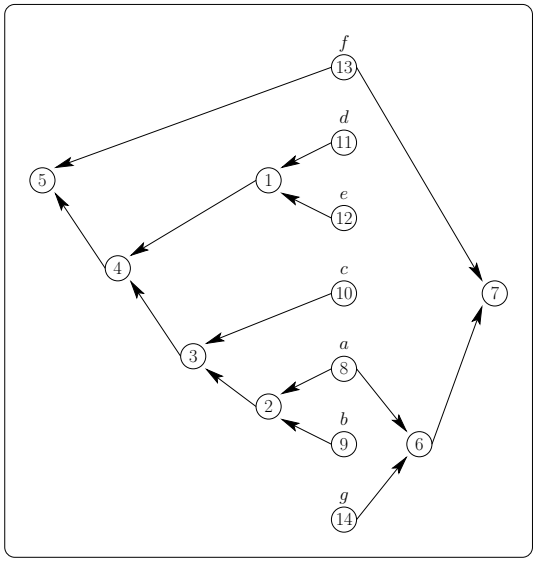

Figure 5.572: Precedence graph associated with the two trees to merge described by Figure 5.561

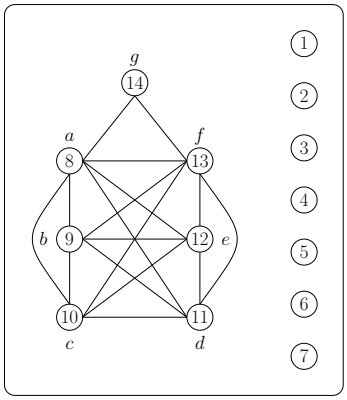

Figure 5.573: Incomparability graph associated with the two trees to merge described by Figure 5.561; the two cliques respectively correspond to the leaves of the two trees to merge.